Parsing: A never ending story

Legacy encodings as

1. Message Text Format (MTF). Over 250 instances of message types using this encoding.

2. Operational Specification for Over the Horizon Targeting (OS-OTG)

3. World Meteorological Organization formats (WMO Metoc)

are not context-free languages and are based on complex lexical syntax.

Parser generators as Lex/Yacc do not work well for these.

Examples, MTF and OS-OTG

Casuality Report (MTF):

```
MSGID/CASREP/CV 63 KITTY HAWK/27//
POSIT/5430N2-04645W9/151615ZAPR86//
CASUALITY/INITIAL-86006/No.2 6L-16 02 GENERATORS//
...
```

PIMTRACK (OS-OTG):

```
MSGID/CTG 81.0/PIMTRACK/0007/JAN
SEC/UNCLASSIFIED
ADDEE/JOTS
PMTRK/TRANSITEX 94-2/GC/75NM/SUR/PHASE 1
LEG/0306000Z9/FEB/94/370000N0/0760000W3
...
LEG/0817010Z7/FEB/94/183000N2/0663000W5
ENDAT
```

RegReg, a parser generator for irregular languages

Used to describe parsers with complex lexical structure.

Based on regular expressions **only**.

The grammar and the tokens are specified using regular expressions.

Built in Scheme (a Lisp dialect).

Used at FNMOC for decoding WMO and AFMAN Metoc messages.

Upcoming publication: Working Conference on Reverse Engineering, November 2003.

Example of parser description

```
(RegReg
  (declare (name example1))
  (macros  (blank "[ \\010\\013]"))
  (level 1
     (w  "[A-Za-z]*{l=[A-Za-z]}")
     (n  "{f=[1-9]}[0-9]*")
     (s  "[-,=]")
     (e  "[.;:!]")
     (b  "{blank}+"))
  (level 2  (line ((+ (? b) (: s w n)) e)))
  (level 3  (text (* line) process-text)))
```

Result is a tagged parse tree

"There are 6 words in this sentence.

A second line."

```
(text
  ((line
      ((((()       (w (("Ther") (l "e"))))
        ((b " ") (w (("ar") (l "e"))))
        ((b " ") (n ((f "6") ())))
        ((b " ") (w (("word") (l "s"))))
        ((b " ") (w (("i") (l "n"))))
        ((b " ") (w (("thi") (l "s"))))
        ((b " ") (w (("sentenc") (l "e"))))) (e ".")))
   (line
      ((((b " ") (w (() (l "A"))))
        ((b " ") (w (("secon") (l "d"))))
        ((b " ") (w (("lin") (l "e")))))
        (e ".")))))
```

RegReg is secure

It is impossible to write a description generating a parser looping forever (denial of service).

RegReg generates deterministic parsers: they never take an exponential amount of time to parse.

Tagging of substrings

RegReg parsers automatically tag any part of a recognized string as specified by the description.

This eliminates extracting substrings out of recognized strings.

The substrings can be stored in a database table, automatically tagged.

Can be used to create XML-like attribute and element tags.

Complex Schema as in MTF

MTF has its own Schema language (conditional rules, based on values) quite unlike XML Schema.

RegReg can be a component of a generator of MTF parsers and validators.

RegReg link:

http://www.metnet.navy.mil/~latendre

END

Thank you

Questions?